Knowledge and Information Systems (2025) 67:5965–5998
https://doi.org/10.1007/s10115-025-02394-6

RESEARCH

A synthetic over-sampling method with minority and majority classes for imbalance problems

Hadi A. Khorshidi[1,2] · **Uwe Aickelin**[1]

Received: 8 December 2023 / Revised: 28 November 2024 / Accepted: 7 March 2025 /
Published online: 12 April 2025
© The Author(s) 2025

Abstract
Class imbalance is a substantial challenge in classifying many real-world cases. Synthetic over-sampling methods have been effective to improve the performance of classifiers for imbalance problems. However, most synthetic over-sampling methods generate synthetic instances within the convex hull formed by the existing minority instances as they only concentrate on the minority class and ignore the vast information provided by the majority class. They also often do not perform well for extremely imbalanced data, as fewer minority instances mean less information with which to generate synthetic instances. Moreover, existing methods that generate synthetic instances using the majority class distributional information cannot perform effectively when the majority class has a multi-modal distribution. We propose a new method to generate diverse and adaptable synthetic instances using Synthetic Over-sampling with Minority and Majority classes (SOMM). SOMM generates synthetic instances diversely within the minority data space. It updates the generated instances adaptively to the neighbourhood including both classes. Thus, SOMM performs well for imbalance problems. We examine the performance of SOMM for binary multi-class imbalance classification problems for different imbalance levels. The empirical results and nonparametric statistical testing show the superiority of SOMM compared to existing methods. We also discuss the strengths and limitations of SOMM through visualisations.

Keywords Imbalanced data · Synthetic over-sampling · Classification · Machine learning

1 Introduction

Interesting phenomena for prediction are usually rare, high-risk, or undesirable which by their nature create a minority class. Typical examples for such phenomena are software failures [1], cancer patients [2], post-surgery mortality [3], face recognition [4], image classification

✉ Hadi A. Khorshidi
hadi.khorshidi@unimelb.edu.au

[1] School of Computing and Information Systems, The University of Melbourne, Parkville, VIC, Australia

[2] Melbourne School of Population and Global Health, The University of Melbourne, Parkville, VIC, Australia

[5] and online fraud [6]. In these examples, the number of instances is not equally distributed between classes and at least one class has substantially different frequency. This situation is known as class imbalance and is one of the main challenges for classification algorithms [7]. Standard classifiers perform under the assumption that classes are approximately balanced. When one class has a considerably higher number of instances, the trained classification model will be biased towards that class due to a high prior probability. This bias makes the classification model perform poorly distinguishing between classes, while the accuracy may remain high. The same reduction in performance happens when one class has low number of instances and a low prior probability [8].

Several methods have been developed to address the class imbalance issue given its importance in data mining. These methods can generally be grouped into two categories: classifier-design and data-balancing. The classifier-design category aims to adjust the training process of the classifiers by considering the minority class without changing the training data set. Developing cost-sensitive classifiers is one approach in this category. Cost-sensitive classifiers assign different costs to misclassification errors of different classes as the undesirability of the misclassification varies from one class to another. For instance, misclassifying a patient whose risk of mortality after surgery is high as a low-risk patient is more critical than the other way around. So, the classifier minimises the cost-based weighted misclassification error instead of minimising the regular misclassification error [9, 10]. Another approach is thresholding that alters the decision boundary in classification [11].

The data-balancing category includes methods that re-sample the training data set to balance the classes [12]. Random under-sampling and random over-sampling are the most basic methods for re-sampling. The main shortcomings of these methods are the loss of valuable information and increasing the chance of overfitting, respectively [13, 14]. Another shortcoming of random over-sampling is adding a number of identical instances into the training data which is a challenge for classifiers [7]. Random over-sampling can be performed in a focused way, i.e. replicating minority instances that are close to the boundary between classes. However, the problem of adding identical instances still holds. In addition, focused over-sampling results in the construction of smaller and more specific decision regions [13]. To overcome these shortcomings, synthetic over-sampling methods have been introduced. These methods have been effective to improve the performance of classifiers when class imbalance exists [14, 15].

2 Related work

The Synthetic Minority Over-sampling Technique (SMOTE) [16] is the most commonly used method and generates synthetic instances using nearest neighbours of the minority class, which broadens the data space for minority instances. SMOTE has shown success in stabilising the performance of classifiers [17]. Many variants have been developed to improve the performance of SMOTE, such as Borderline-SMOTE [18], Adaptive Synthetic Sampling (ADASYN) [19], density-based SMOTE (DBSMOTE) [20], Evolutionary Cluster-Based Synthetic Over-sampling Ensemble (ECO-Ensemble) [21] and Majority Weighted Minority Over-sampling Technique (MWMOTE) [22], known as the SMOTE family [23]. However, as we will explain in the next section, limitations remain for SMOTE family algorithms and other synthetic over-sampling methods, which motivate our study.

Recently, Dablain et al. [24] have proposed a hybrid over-sampling algorithm using SMOTE for Deep Learning (DL) to generate high-quality synthetic images. In another attempt

to improve the performance of DL models for imbalanced and long-tailed data sets, Bellinger et al. [25] propose a method of re-sampling through mini-batches, instance mixing and soft-labels. Also, generative adversarial networks (GANs) have been used to tackle imbalance problems in DL by considering both categories [26].

MAHAKIL [17] is a synthetic over-sampling method that has been introduced recently to overcome the limitations of the SMOTE family, notably by generating more diverse synthetic instances. Inspired by the Chromosomal Theory of Inheritance [27], MAHAKIL generates synthetic instances by pairing existing minority instances and previously generated synthetic instances as parents to generate new instances. Synthetic over-sampling with the majority class (SWIM) [14] has recently been introduced to capture majority class information for synthetic instance generation. SWIM uses Mahalanobis distance [28] to generate synthetic instances that are in the same distance from the mean of the distribution of the majority class as the existing minority instances. In [29], the authors propose a nonparametric version of SWIM using radial basis function instead of Mahalanobis distance. Recently, Shi et al. [30] have proposed an ensemble re-sampling algorithm based on instance concatenation considering both distributions of the original data set and the majority class.

Dai et al. [31] have proposed a Three-line Hybrid Positive Instance Augmentation (THPIA) to generate unlabelled instances from both minority and majority classes to be added into minority class based on distance constraints. Feature Information Aggregation Over-sampling (FIAO) is proposed in [32] to leverage feature importance, feature density, and standard deviation in the over-sampling process to avoid issues related to parameter selection and boundary effects. In [33], Hypergraph identification and Gaussian Distribution Over-sampling (HGDO) is proposed to filter outliers and noisy instances to generate synthetic samples. To address the noise issue in synthetic over-sampling without losing robustness, Li [34] proposes a wrapping over-sampling approach based on sample subspace optimisation (SSO) with accelerated binary particle swarm optimisation (PSO) that can applied on other over-sampling methods. SSO and PSO have been used previously for semi-supervised classification [35, 36].

Multiclass imbalance problems are more challenging than binary imbalance problems, so studies have addressed multiclass problems exclusively with synthetic over-sampling [37]. Mahalanobis Distance-based Over-sampling technique (MDO) [38] generates synthetic instances for each minority class using Mahalanobis distance and distributional information of that minority class within the ellipse contour that passes through the chosen instances of the minority class. Finally, the generated instances are transformed back into the original space. A k-nearest neighbours (k-NN)-based synthetic minority over-sampling algorithm (SMOM) [39] is proposed to generate synthetic instances for each minority class using k-NN of the minority instances through linear interpolation like SMOTE. However, to avoid over-generalisation, SMOM clusters the minority instances using a neighbourhood-based clustering algorithm (NBDOS) which is derived from a density-based clustering method, DBSCAN [40]. NBDOS groups minority instances based on the density of minority instances in their neighbourhood. The instances that are in the regions that are dominated by the minority class are "outstanding instances" and others are "trapped instances".

GMMSampling [41] fits a Gaussian Mixture Model to each minority class to estimate the distribution and generates instances based on the estimated distribution. It prioritises regions with the most unsafe minority classes guided by calculated safe levels. It also undersamples the majority instances to reach a balanced size. Heterogeneous Clustering Ensemble method for Multiple Class-overlap Detection (HCE-MCD) [42] identifies the overlapping instances using genetic algorithm, a combination of heterogeneous clustering techniques and majority voting for multiclass imbalance problems. Then, it eliminates the class-overlap problem in

the multiclass data set. There are few methods that work for both binary and multiclass imbalanced data.

3 Motivations

In this section, we review the limitations of the existing synthetic over-sampling methods. We use two examples (shown in Fig. 1) to illustrate these limitations. Example 1 visualises the situation in which the data space for the majority class is a convex hull, while the minority class consists of two separate regions. Example 2 visualises the situation in which the data space of the majority class is bi-modal, and the region of the minority class is a convex hull located between two high-density regions of the majority class.

One of the limitations of SMOTE is that it generates synthetic instances within the convex hull formed by the minority instances existing in the training data set. This can lead to generating synthetic instances in the regions belonging to the majority class (over-generalisation). Figure 1a shows how SMOTE generates instances in the data space of the majority class, especially in the region with higher density that should be avoided to prevent reducing the performance of classifiers.

SMOTE and its variants use similar procedures to generate synthetic instances using nearest neighbours. Generating synthetic instances based on nearest neighbours results in having less diverse synthetic data [17] and complications in covering the data space of the minority class [43]. Keeping diversity in re-sampling methods improves the performance of classifiers for imbalanced data [44, 45]. By diversity, we mean to avoid generating identical (or very similar) synthetic instances which leads to a smaller decision region. In other words, synthetic instances should fully cover the region that dominated by the minority class [46]. Figure 1a, b shows this limitation of SMOTE as the synthetic instances are linear interpolations of existing minority instances.

Figure 1c, d shows the performance of MAHAKIL when generating synthetic instances for both examples. Figure 1c shows that MAHAKIL performs better than SMOTE by generating

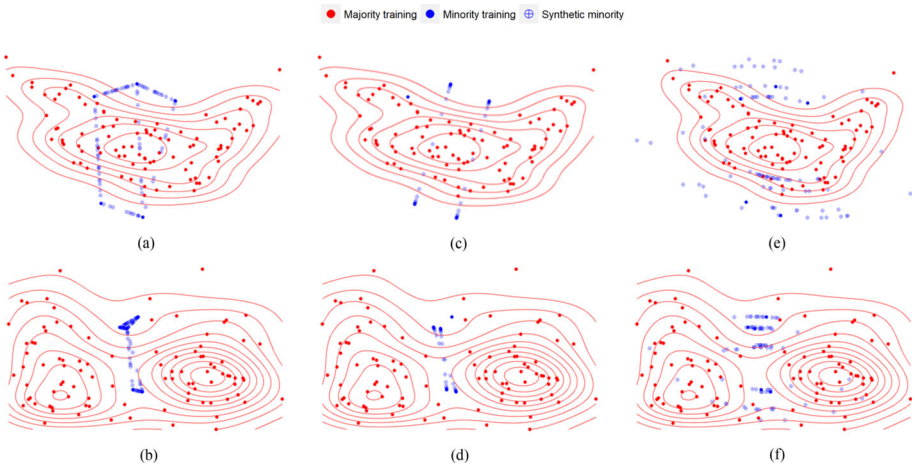


Fig. 1 Two examples for illustration of synthetic generation using SMOTE (**a**, **b**), MAHAKIL (**c**, **d**) and SWIM (**e**, **f**)

fewer instances in the data space of the majority class. However, both examples show that MAHAKIL cannot generate diverse synthetic instances when the number of instances in the minority class is low. This is because MAHAKIL can only assure the diversity when there are enough and diverse parents to select from [45, 47].

Another limitation of both SMOTE and MAHAKIL is that they only use the existing minority instances to generate synthetic instances, ignoring the vast information provided by the majority class. Figure 1e shows SWIM can successfully generate synthetic instances that avoid the higher density region of the majority class even though some of the synthetic instances are in its data space. However, SWIM cannot always avoid generating synthetic instances in the higher density region of the majority class when its distribution is bi-modal (Fig. 1f). This is because the mean of the distribution is not located in the higher density region of the bi-modal distribution. In addition, the synthetic instances generated by SWIM are not diversely distributed in the data space of the minority class. SWIM's algorithm involves the computation of matrix inverses and square roots of those inverses. This leads to failing the computation if the features are linearly dependent. To overcome this issue, there is a need for pre-processing to proceed with SWIM's algorithm on the subspace with independent features [14].

Most over-sampling methods, including the SMOTE family and MAHAKIL, do not perform well in cases of extreme imbalance, because they rely on minority instances to generate synthetic instances. Thus, the fewer the minority instances, the less information is provided to generate synthetic instances. SWIM is a powerful method to generate synthetic instances when the minority instances in the training data set are limited. However, its performance deteriorates in comparison with existing synthetic over-sampling methods as the number of minority instances increases [14]. Dai et al. [48] address the issue of class-overlap in imbalance data which makes classification more challenging. They have expanded the research on class-overlap for multiclass imbalance problems [42].

Synthetic over-sampling methods can widen the data space of the minority class by generating synthetic instances in the regions of the majority class. This is not crucial when the number of these synthetic instances is few. However, if the generated instances either are in higher density regions of the majority class or form a new decision region, they can provide misleading information for classifiers.

4 Contributions

To address the limitations and concerns mentioned in the motivation section, we define our research questions (RQs) as follows:

RQ1: How to generate synthetic instances adaptable to the data space of the majority class for cases such as non-convex minority data space?
RQ2: How to generate synthetic instances diversely to cover the data space of the minority class?
RQ3: Given the data space of both minority and majority classes, where should the synthetic instances be generated?
RQ4: Is it possible to have a synthetic over-sampling method that can perform appropriately for high imbalance (around 5%), extreme imbalance (1% and less) and absolute imbalance (few minority instances) data?

In this study, we aim to propose a new method of generating synthetic instances to address the defined research questions. The proposed method is Synthetic Over-sampling with the

Minority and Majority classes (SOMM). SOMM does not use the nearest neighbours of the minority class to generate synthetic instances as the SMOTE family does. Thus, SOMM can generate diverse synthetic instances within the data space of the minority class. However, SOMM considers the nearest neighbours from both minority and majority classes when it locates the synthetic instances by maintaining the diversity. As a result, SOMM remains adaptable to the data space of the majority class and non-convexity of the data space of the minority class. These properties allow SOMM to improve classifiers for class-overlap [48] and multiclass imbalance problems [49]. SOMM can generate diverse synthetic instances for each minority class that are accountable to other classes. Attention in learning from imbalanced data has mainly been directed to binary problems, while multiclass problems are relatively more challenging to deal with. Normally, the methods that work for binary problems do not perform correctly for multiclass problems. However, in this paper, we show that SOMM can perform effectively for both binary and multiclass problems. SOMM is effective to generate synthetic minority instances for the cases that minority instances in the training data are very low.

We empirically examine the ability of SOMM to generate diverse synthetic instances to cover the minority space using 6 synthetic data sets. We also validate the performance of SOMM in comparison with existing recent methods using 26 benchmark data sets, including 6 synthetic and 20 real data sets, in high, extreme and absolute situations for binary imbalance problems, and 10 data sets for multiclass imbalance problems. To encourage further take-up, we also ensure that SOMM is formulated intuitively and relatively simple and can be applied to different types of data with no pre-processing step.

5 Method

In this section, we describe the details of our SOMM algorithm (Algorithm 1[1]) for binary imbalance problems. This algorithm can be expanded for multiclass imbalance problems by targeting one of the minority classes as the minority class and all other classes as the majority class. We then discuss the advantages of SOMM in comparison with other synthetic over-sampling methods.

5.1 Algorithm

The algorithm aims to generate synthetic instances via addressing RQs 1–3. The generated synthetic instances diversely cover the data space of the minority class. In addition, the algorithm is adaptable to the neighbourhood. This algorithm adapts to the neighbourhood by taking into account both minority and majority data spaces when it confirms the placement of the generated synthetic instance. Consequently, SOMM would perform well when the number of minority instances is extremely low in comparison with majority instances. We describe the five steps of SOMM in details as follows. The training data set is indicated as D, and the number of synthetic instances that need to be created to balance the training data set is denoted by N_S.

[1] The codes are provided at https://github.com/hadi1453/Synthetic-Over-sampling-with-the-Minority-and-Majority-classes-for-imbalance-problems/blob/main/SOMM.R.

5.1.1 Normalise the data set

Before generating the synthetic instances, we transform the data set so that all features are ranged between zero and one. This transformation enables the application of Euclidean distance (as widely used in the literature) in the fourth step to update the generated synthetic instances properly. The z-score normalisation does not work, as SOMM does not consider distributional information. (This has been investigated empirically as well.) Let $\mathrm{Max_D}$ and Min_D be the vectors of maximum and minimum values for features, respectively. Thus, the normalised data set (D_{norm}) is obtained using (1).

$$D_{norm} = (D - Min_D) \div (Max_D - Min_D) \tag{1}$$

5.1.2 Set feature bounds

After the transformation in the first step, we separate D_{norm} into two subsets of majority and minority classes. Let A_{norm} and B_{norm} be the transformed subsets of majority and minority, respectively. Then, for each feature in B_{norm}, we set maximum and minimum values as upper (u_f) and lower (l_f) bounds.

5.1.3 Generate synthetic instances

We randomly generate a value between l_f and u_f for each feature. Thus, we generate synthetic instances (s) as vectors where all elements lie within feature bounds. Consequently, the generated instances can be diversely spread across the data space of the minority class (addressing RQ2).

5.1.4 Update the generated instances

Although we generate the instances inside the feature bounds, they may be placed outside the data space of the minority class. For instance, if the minority class is not convex, the space bounded by the feature bounds may overlap the data space of the majority class. So, some of the synthetic instances that are generated in step 3 may breach the border and end up in the regions of the majority class.

We address this issue by investigating the neighbourhood of the generated instance. Once an instance (s) is generated, we find k nearest neighbours of that generated instance. These neighbours could be from both majority and minority classes. We apply the three following rules to prevent the widening of the data space of the minority class by generating synthetic instances in the regions of the majority class.

1. If all the neighbours belong to the majority class, we remove that generated instance (addressing RQ1).
2. If the condition of rule (1) is not met, we order the neighbours based on their distance from the generated instance. If the closest neighbour belongs to the minority class, we keep that generated instance and update it as S_u (updated instance).
3. If the condition of rule (2) is not met, we cut the ordered set of neighbours at the point of the closest neighbour from the minority class. After that, we would have a subset (nn) of the ordered set that includes a set of the closest neighbours from the majority class (nn_A) and ending with the closest neighbour from the minority class (nn_B).

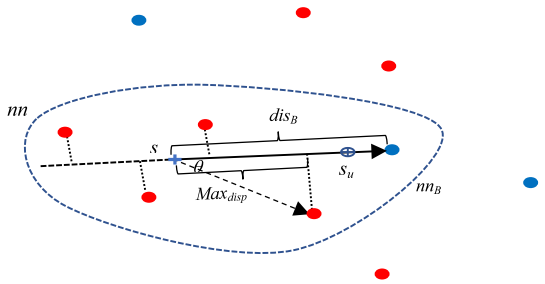


Fig. 2 An illustrative example for updating the generated instances; red and blue circles are instances from majority and minority classes, respectively, + denotes the generated instance; and ⊕ is the updated generated instance

We assimilate the generated instance to the minority class by moving from the neighbourhood of the majority class towards nn_B. To do this, we need to define the direction and the magnitude of the movement. The direction (dir) is the vector from the generated instance to nn_B. If the magnitude (m) equalises with the distance between nn_B and the generated instance (dis_B), nn_B would be replicated. As we want to maintain the diversity in the over-sampled data, we avoid replicating nn_B. We therefore set the magnitude such that the generated instance moves away from the neighbourhood of the majority class (addressing RQ3). To do this, we need to project the neighbours in nn_A onto dir. First, we calculate the $Cosine$ of the angle (θ) between dir and the vectors from the generated instance to the neighbours in nn_A (dir_A) using (2). Then, we find the projected distance (dis_P) of each neighbour in nn_A onto dir using the distance of each neighbour from the generated instance (dis_A) as (3).

$$Cosine\ \theta = (dir_A \cdot dir) \div (|dir_A|_2 \times |dir|_2) \tag{2}$$

$$Dis_P = dis_A \times Cosine\theta \tag{3}$$

where $|\ |_2$ indicates the norm 2 that calculates the size of each vector. Also, $Cosine\ \theta$ and dis_P are sets with the length of nn_A. We obtain the maximum value in dis_P (Max_{disp}). Then, we randomly select a number between Max_{disp} and dis_B to set as the magnitude. Lastly, the generated instance is updated using (4).

$$s_u = s + dir \times m \tag{4}$$

Figure 2 shows the process of updating the generated instance described in step 4 of the SOMM algorithm. The generated instance + is updated as ⊕ by moving towards the minority instance (blue circle) with the extent of magnitude between Max_{disp} and dis_B.

5.1.5 Transform back to the original space

As the final generated instances (s_u) are still in the normalised space, we need to transform them into the original space using (5). They can then be added to the training data set for constructing classifiers.

$$s_{SOMM} = s_u \times (Max_D - Min_D) + Min_D \tag{5}$$

where s_{SOMM} are the synthetic instances generated by SOMM method for over-sampling the minority class.

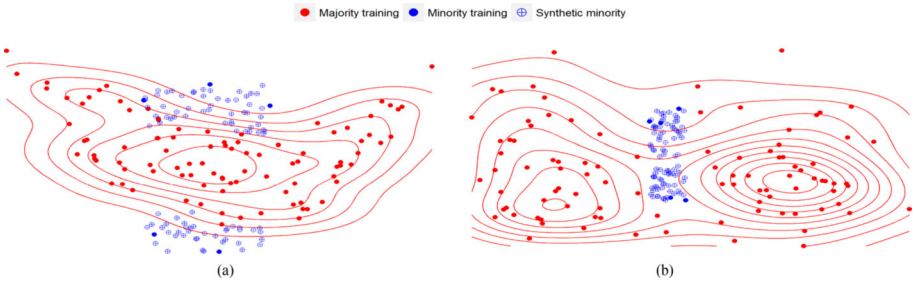


Fig. 3 Two examples for illustration of synthetic generation using SOMM

5.2 Comparative discussion for binary imbalance problems

In this section, we compare SOMM with other studies in terms of generating synthetic instances. We also apply SOMM on two examples used in Fig. 1. Figure 3 shows how SOMM performs on these examples. Ideally to construct a robust classifier, the generated synthetic instances should correspond with the real data space of the minority class. In Fig. 3a, we can see that SOMM keeps the non-convex space of the minority class. In both figures, there are no synthetic instances generated in the higher density regions of the majority class. Moreover, SOMM can generate diverse synthetic instances to cover the data space of the minority class even when the number of existing instances is low.

SWIM's performance deteriorates when the majority class is not a convex hull (Fig. 1f) due to its reliance on distributional information of the majority space. SOMM, however, can generate synthetic instances that are not in the higher density regions of the majority class even if there is a bi-modal space for the majority class (Fig. 3b). Also, SOMM generates synthetic instances diversely across the minority data space, while SWIM only generates synthetic instances in the same hyper-elliptical density contour around the majority class. MAHAKIL may widen the data space of the minority class mistakenly into the majority space. However, SOMM considers the information of the neighbourhood from both majority and minority classes to avoid generating instances within the majority space. MAHAKIL fails to generate diverse instances when there are few minority instances (Fig. 1), whereas even in these cases, SOMM can cover the minority space diversely by generating synthetic instances.

Although some of the methods in the SMOTE family incorporate the information of the majority class in cleaning the generated instances, they still rely on generating synthetic instances through SMOTE [14]. Thus, the limitations of generating non-diverse synthetic instances within the convex hull of the minority instances still hold. In comparison with Schur Decomposition Class-overlap Under-sampling (SDCU) [48], SOMM can generate effective synthetic instances accountable to class-overlap without under-sampling the majority class.

5.3 Comparative discussion for multiclass imbalance problems

In this section, we compare SOMM with MDO [38], SMOM [39], GMMSampling [41] and HCE-MCD [42] as comparable synthetic over-sampling methods for multiclass imbalance problems. All these methods use computationally intensive algorithms. MDO examines the neighbourhood of every single instance in all minority classes to choose appropriate instances and assign weights. SMOM also examines the neighbourhood of all instances in minority

classes to find their cluster. Then, all trapped instances are examined using new rules to assign weights to be involved in generating synthetic instances. GMMSampling needs to check every instance in the majority classes to undersample. HCE-MCD uses an evolutionary algorithm to select a combination of heterogeneous clustering techniques with the best fitness function amongst a pool of clustering methods. SOMM, on the other hand, is formulated relatively simply.

SMOM needs eight pre-defined parameters. HCE-MCD needs 4 parameters, MDO needs two, but SOMM just needs one pre-defined parameter. In the following sections, we describe the experiments, present the results, and discuss the findings for both binary and multiclass imbalance problems.

5.4 Runtime complexity

SOMM relatively has low runtime complexity. It requires finding the k number of nearest neighbours of each generated instance. If m shows the number of generated instances, the complexity is as $\mathcal{O}(km)$. It should be noted that m might be bigger than the number of synthetic instances needed, because some of the generated instances could be removed due to the first rule of updating the generated instances (Sect. 5.1.4). So, the runtime of SOMM is not necessarily dependent on the size of a data set. However, it depends on how much minority class and majority class-overlap.

6 Experiment results for binary imbalance problems

6.1 Experimental design

We design experiments to compare the performance of SOMM with other synthetic over-sampling methods for binary imbalance problems. We choose SMOTE as the most used synthetic over-sampling method, and its family methods such as ADASYN, MWMOTE and DBSMOTE. We also choose MAHAKIL and Diversity-based Instance-Wise Over-sampling (DIWO) [45] as two recently introduced methods which focus on diversity and SWIM which uses majority distribution to generate synthetic instances. Figure 4 shows the framework of

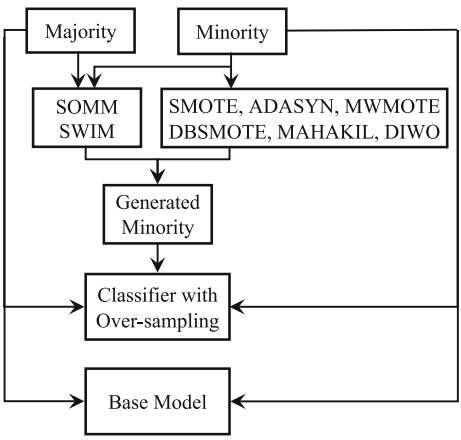


Fig. 4 The framework of experimental design

experimental design using minority and majority information via synthetic over-sampling methods to construct classifiers.

We implement SMOTE, ADASYN and DBSMOTE methods using the smotefamily R package [50] and MWMOTE using the imbalance R package [51]. The parameter values for SMOTE is $k = 5$, for ADASYN is $k = 5$, for DBSMOTE are $MinPts = 4$ and $eps = 10\%$, and for MWMOTE are $k1 = 5$, $k2 = 3$, $k3 = 3$, $C_f(th) = 5$, $CMAX = 2$ and $Cp = 3$. For SWIM, DIWO and MAHAKIL, we develop the algorithm as described by the authors in the papers [14, 17, 45], and we set the parameters for SWIM as $\alpha = 2$, for DIWO as $n = 50$, $g = 20$, $r = 30$, $v = 0.9$, $c = 1$, and no parameter for MAHAKIL.

6.1.1 Diversity

In this section, we want to examine the performance of SOMM in generating diverse synthetic instances within the minority data space (examining RQ2). In previous works, this property has been discussed using visualisation [14, 43] as the data space of the minority class is not normally accessible. However, we design an experiment using synthetic data sets (SDs) to measure how well each over-sampling method can cover the minority space. We create 6 data sets that represent different situations such as non-convex minority space, bi-modal majority, linearly separable classes, and overlapped classes (examining RQ1).

Figure 5 shows these data sets. In SD1, the majority space is a convex hull, while the minority space is non-convex and divided by the majority space. Two classes have small overlaps. In SD2, the majority space is bi-modal, while the minority space is located between high-density regions of the majority class as a convex hull. The minority space does not overlap with the high-density regions of the majority class. In SD3, both minority and majority spaces are convex and separated linearly from each other. In SD4, the minority instances are inside the majority space and both spaces are convex. In SD5, the minority space is non-convex and is within the majority space. In SD6, both spaces are convex, and the minority space is within the majority space and overlaps with high-density region of the majority instances.

As the aim is to generate synthetic instances that fully cover the region dominated by the minority class, we measure the diversity of synthetic instances in a way that they uniformly

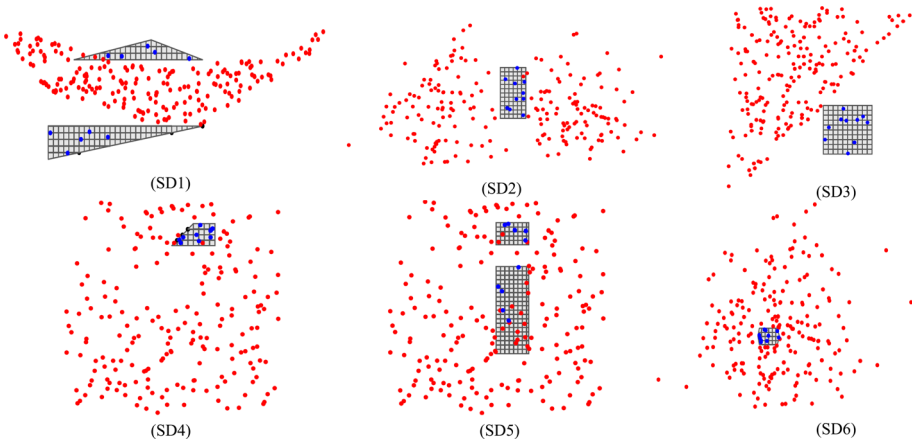


Fig. 5 Synthetic data sets

cover the minority space. Inspired by spatial coverage measures [52], we divide the minority space into cells. Then, we count the number of cells that do not contain any synthetic instances (not covered cells, NCC). We measure the coverage diversity (CovDiv) using (6) by calculating the ratio of cells contain synthetic instances.

$$CovDiv = 1 - \frac{NCC}{number\ of\ cells} \tag{6}$$

The CovDiv not only measures the diversity, but also ignores the synthetic instances that located outside the minority space. It should be mentioned that the CovDiv score is to illustrate our definition of coverage and diversity, and it is not necessarily associated with the classifiers' performance.

For this experiment, we keep the number of majority instances as 100 but set the number of minority instances at three different values of 6, 10 and 20 to examine the methods under different situations. We set the minimum number of minority instances as 6 because SMOTE and MWMOTE cannot work especially when $K = 5$ as the optimal number of neighbours [14]. Also, MAHAKIL cannot work if the minority instances are less than 4. We repeat each experiment 30 times to check different possibilities of minority instances within their space for an unbiased evaluation. We also conduct Mann–Whitney test, a non-parametric statistical test, to examine the significance of the differences across over-sampling methods. We use Mann–Whitney tests as we cannot ensure the normality of g-mean values.

6.1.2 Classification

In this section, we examine the performance of over-sampling methods in classification tasks. We use 6 synthetic data sets and 20 real benchmark data sets[2] (listed in Table 1) in this experimental design. We select these data sets as they can cover various ranges of size and dimension to provide a fair base of comparison. In addition, these data sets have enough minority instances to create training data sets with different levels of class imbalance as required for RQ4. For SDs, we set the number of majority instances as 1000, and minority instances as 100 to have enough instances for experiments in different levels. We create three different training data sets with 5% (high), 1% (extreme) imbalance ratios and only 6 minority instances (absolute imbalance).

We randomly divide the data sets into training and testing subsets by ratios of 75% and 25%, respectively. We choose the ratio of 75% for training subset to have more minority instances in the training data. We set the minimum number of minority instances as 8 for data with 1% imbalance ratio to differentiate between extreme and absolute imbalance levels. Then, we randomly down-sample the minority instances in the training subset to reach the desired imbalance levels. This process is repeated 30 times to cover different combinations in our performance evaluation and provide a possibility for conducting statistical tests.

Once the training data are created, the synthetic over-sampling methods are implemented to balance the majority and minority classes. Then, the final data set is used to construct classifiers. We also construct classifiers without over-sampling as base models (as shown in Fig. 4). We use seven classifiers to provide a comprehensive investigation. These classifiers are Logistic Regression (LR), Naïve Bayes (NB), Decision Trees (CART), k-nearest neighbour (k-NN), Neural Networks (NN) with single hidden layer, Random Forest (RF) and Support

[2] https://archive.ics.uci.edu/ml/datasets.php; https://web.stanford.edu/~hastie/ElemStatLearn/datasets/; https://machinelearningmastery.com/imbalanced-classification-model-to-detect-oil-spills/.

Table 1 Data sets for binary imbalance problems

Data set	Name	Dimension	Size	Ratio*	Data set	Name	Dimension	Size	Ratio
D1	Wisconsin	9	683	65–35	D11	Vehicle Bus	18	846	74–26
D2	Waveform 0	21	1800	67–33	D12	Pima	8	768	65–35
D3	Vowel 0	10	990	91–09	D13	Wine White High§ Quality	11	4898	78–22
D4	Vowel 1	10	990	91–09	D14	Wine White Low vs High	11	1243	85–15
D5	Vowel 2	10	990	91–09	D15	Wine Red Low§ Quality	11	1599	96–04
D6	Vowel 10	10	990	91–09	D16	Wine Red Low vs High	11	280	77–23
D7	Poker 7–9 vs 6	10	1669	85–15	D17	Thoracic Surgery	24	470	85–15
D8	KDD Control	60	600	83–17	D18	Norm Ring	20	5532	66–34
D9	Oil	49	937	95–05	D19	Spambase	57	4601	61–39
D10	Forest Cover 3	54	3000	95–05	D20	Abalone 9–18	9	2676	68–32

*Ratio denotes the percentages of majority-minority instances in the original data set
§High refers to the quality values of 7 and above; Low refers to the quality values of 4 and less

Vector Machine (SVM) with radial kernel. We implement all the classifiers using R functions[3] with their default settings.

The constructed classifiers are applied on the test data sets to examine their performance. We use the geometric mean of true positive rate (TPR) and true negative rate (TNR), where positive and negative corresponds to majority and minority classes, respectively, to measure the performance. This measure is known as the g-mean and is calculated as (7).

$$g{-}mean = \sqrt{TPR \times TNR} \tag{7}$$

G-mean is an appropriate measure for class imbalance problems as TPR and TNR are calculated separately for each class. Also, g-mean is a recommended measure amongst others for general evaluation in binary classification [53].

We compare the synthetic over-sampling methods and the base model using the average of g-mean for the best-performing classifier. Then, we conduct the statistical testing on the difference of SOMM with other alternative methods using Friedman and Nemenyi post hoc tests [54]. In the Friedman test, the null hypothesis is that the methods' performance is significantly different, and it will be rejected if all methods perform the same. The Nemenyi test is a nonparametric post hoc test that compares the performance of multiple methods across multiple data sets through their ranking. It also can visualise the critical difference (CD) based on a chosen significance level. These tests have been used in several studies to compare between the proposed over-sampling method and alternatives [42, 48]. Before conducting the comparison, we need to find an appropriate optimal value for the only parameter of SOMM to be used in generating synthetic instances. We check three different values of 5, 10 and 15 for this parameter.

6.2 Results

We obtain experimental results for 26 data sets over three different imbalance levels using seven classifiers with 30 repeats.

6.2.1 Setting the parameter

We need to define the number of nearest neighbours of the generated instance (k) described in step 4. We conduct a preliminary investigation on all 20 data sets[4] to find the best parameter value for SOMM amongst k of 5, 10 and 15. We compare these settings using g-mean based on the best-performing classifier. The results show that $k = 15$ outperforms the others for more data sets within each imbalance level. Figure 6 shows the success percentage of each setting in total for the three levels.

[3] glm() for LR, naiveBayes() for NB, rpart() for CART, knn() for k-NN, nnet() for NN, randomForest() for RF, svm() for SVM.

[4] The partitions are independent from 30 repeats that have been used for comparative experiments. We also used these partitions to find the best parameters for alternative methods.



Fig. 6 The number of data sets on which each parameter performs the best in all three imbalance levels

6.2.2 Diversity comparison

Table 2 presents the results of measuring the coverage diversity for over-sampling methods across 6 synthetic data sets (SDs) and three different numbers of minority instances. We report the mean of CovDiv measure and its standard deviation (SD) over 30 repeats (mean ± SD). The greatest average value is indicated as bold, and if the value is statistically significant, it is indicated as italic.

The results show that SOMM in 16 out of 18 combinations has the greatest value and in 12 of them is statistically greater than others. Only in two cases, SD4 and SD6 with 6 minority instances, SWIM can provide better diverse coverage than SOMM (not statistically significant). This shows SWIM can generate more diverse synthetic instances within the minority space when the minority space is convex and inside the majority space and the number of minority instances is low. Otherwise, SOMM provides superior diversity coverage within the minority space in different situation including non-convex minority space (Addressing RQs 1 & 2).

6.2.3 Classification comparison

We summarise the results by reporting the g-mean averages for the best-performing classifier, i.e. the best g-mean value amongst seven classifiers, using synthetic over-sampling methods and the base model for both synthetic (SD1-6) and real (D1-20) data sets and each imbalance level in Tables 3, 4 and 5. The bold number indicates the best performance, and the italic number indicates the second-best performance.

Table 3 shows that SOMM performs the best for synthetic data sets as it achieves the highest values for three data sets. SOMM performs competitive for other SDs as well. By considering real data sets, MAHAKIL is superior in comparison with other methods by achieving the best performance in 6 data sets. After that, SOMM and ADASYN perform the second-best with having the best g-mean value for 5 data sets. When we look at all 26 data sets, SOMM and MAHAKIL equally stand at the first rank for the high imbalance level (5%).

Table 4 demonstrates that SOMM has the best performance amongst synthetic over-sampling methods for synthetic data sets as it has the highest values for four data sets. For real data sets, SOMM outperforms other methods by having superiority for 11 data sets. SWIM comes the second by having the best g-mean value in 6 data sets. By considering all 26 data sets, SOMM stands superior for the extreme imbalance level (1%).

Table 5 shows that SOMM performs the best for synthetic data sets as it achieves the highest values for four data sets. By considering real data sets, SOMM is superior by being the best for 7 data sets. The second-best is SWIM by achieving the best g-mean value for 6 data sets. Also, by looking at all 26 data sets, SOMM stands out amongst other methods for the absolute imbalance level (6 minority instances).

The overall figures across all three imbalance levels demonstrate the superiority of SOMM in comparison with other methods as being the best for all three levels. The performance of MAHAKIL drops significantly between the high imbalance level and extreme and absolute levels. SWIM's performance deteriorates in comparison with other methods by increasing the number of minority instances. SOMM shows a more consistent performance across all three imbalance levels (addressing RQ4).

Figure 13 (supplementary materials) presents box plots of the ranks of each synthetic over-sampling method on three imbalanced levels. These plots show the superiority of SOMM on highly imbalanced data. Tables 9–15 (supplementary materials) provide additional comparative results based on each classifier for absolute imbalanced level (minority size 6). Based on

Table 2 The CovDiv results for over-sampling methods

Data set	#	SOMM	SMOTE	MAHAKIL	SWIM	ADASYN	DBSMOTE	MWMOTE	DIWO
SD1	6	**0.361 ± 0.087**	0.289 ± 0.056	0.157 ± 0.035	0.339 ± 0.057	0.252 ± 0.054	0.175 ± 0.046	0.191 ± 0.057	0.275 ± 0.062
SD1	10	***0.438 ± 0.069***	0.356 ± 0.052	0.252 ± 0.045	0.374 ± 0.062	0.318 ± 0.049	0.232 ± 0.048	0.250 ± 0.049	0.344 ± 0.059
SD1	20	***0.479 ± 0.063***	0.443 ± 0.044	0.359 ± 0.041	0.383 ± 0.063	0.361 ± 0.052	0.286 ± 0.044	0.331 ± 0.042	0.389 ± 0.059
SD2	6	***0.707 ± 0.162***	0.564 ± 0.126	0.311 ± 0.070	0.591 ± 0.121	0.559 ± 0.128	0.360 ± 0.092	0.463 ± 0.158	0.571 ± 0.134
SD2	10	***0.857 ± 0.101***	0.661 ± 0.110	0.499 ± 0.060	0.687 ± 0.085	0.683 ± 0.112	0.455 ± 0.079	0.555 ± 0.114	0.683 ± 0.101
SD2	20	***0.916 ± 0.064***	0.759 ± 0.087	0.713 ± 0.082	0.707 ± 0.068	0.537 ± 0.125	0.525 ± 0.082	0.671 ± 0.092	0.791 ± 0.078
SD3	6	**0.717 ± 0.170**	0.557 ± 0.117	0.309 ± 0.062	0.703 ± 0.110	0.567 ± 0.125	0.373 ± 0.093	0.448 ± 0.142	0.545 ± 0.125
SD3	10	***0.856 ± 0.098***	0.656 ± 0.097	0.485 ± 0.070	0.744 ± 0.089	0.684 ± 0.109	0.455 ± 0.083	0.510 ± 0.154	0.684 ± 0.110
SD3	20	***0.923 ± 0.053***	0.729 ± 0.096	0.720 ± 0.079	0.799 ± 0.070	0.715 ± 0.095	0.546 ± 0.080	0.514 ± 0.103	0.775 ± 0.076
SD4	6	0.605 ± 0.173	0.493 ± 0.113	0.276 ± 0.050	**0.629 ± 0.136**	0.452 ± 0.116	0.324 ± 0.096	0.395 ± 0.113	0.491 ± 0.118
SD4	10	**0.722 ± 0.141**	0.540 ± 0.086	0.415 ± 0.074	0.721 ± 0.105	0.502 ± 0.114	0.378 ± 0.073	0.437 ± 0.089	0.550 ± 0.089
SD4	20	**0.826 ± 0.089**	0.650 ± 0.061	0.633 ± 0.063	0.806 ± 0.072	0.441 ± 0.092	0.474 ± 0.072	0.577 ± 0.073	0.686 ± 0.071
SD5	6	***0.537 ± 0.097***	0.442 ± 0.084	0.193 ± 0.037	0.424 ± 0.071	0.437 ± 0.094	0.263 ± 0.062	0.269 ± 0.107	0.412 ± 0.093
SD5	10	***0.599 ± 0.082***	0.499 ± 0.073	0.328 ± 0.044	0.496 ± 0.064	0.464 ± 0.071	0.299 ± 0.061	0.400 ± 0.067	0.512 ± 0.065
SD5	20	***0.608 ± 0.057***	0.569 ± 0.067	0.545 ± 0.070	0.496 ± 0.053	0.449 ± 0.073	0.381 ± 0.063	0.444 ± 0.065	0.549 ± 0.057
SD6	6	0.635 ± 0.177	0.479 ± 0.125	0.269 ± 0.056	**0.655 ± 0.100**	0.483 ± 0.127	0.374 ± 0.077	0.397 ± 0.108	0.503 ± 0.142
SD6	10	***0.812 ± 0.083***	0.608 ± 0.085	0.476 ± 0.089	0.741 ± 0.085	0.593 ± 0.125	0.423 ± 0.072	0.516 ± 0.090	0.651 ± 0.110
SD6	20	***0.931 ± 0.059***	0.745 ± 0.079	0.695 ± 0.080	0.840 ± 0.071	0.564 ± 0.130	0.533 ± 0.066	0.656 ± 0.082	0.789 ± 0.066

Table 3 The best *g*-mean value obtained by synthetic over-sampling method for high imbalance level (5%)

Data set	Base	SOMM	SMOTE	MAHAKIL	SWIM	ADASYN	DBSMOTE	MWMOTE	DIWO
SD1	0.951	*0.972*	*0.972*	**0.979**	0.936	0.968	0.956	0.951	0.931
SD2	0.903	**0.984**	0.973	*0.980*	0.947	0.981	0.972	0.979	0.978
SD3	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
SD4	0.939	*0.981*	*0.981*	0.979	0.977	0.978	0.974	0.978	**0.982**
SD5	0.645	**0.942**	0.937	0.931	0.891	0.932	0.934	0.934	*0.938*
SD6	0.622	*0.935*	0.923	*0.935*	**0.939**	*0.935*	0.918	0.919	*0.935*
D1	0.963	0.964	0.964	0.964	0.964	**0.967**	0.960	*0.965*	0.964
D2	0.786	*0.843*	0.840	**0.844**	*0.843*	0.817	0.798	0.836	0.782
D3	0.842	*0.974*	0.970	0.972	0.961	0.964	0.953	0.967	**0.982**
D4	0.563	0.945	0.944	*0.952*	0.911	**0.958**	0.928	0.949	0.935
D5	0.557	0.952	0.943	0.948	0.909	**0.958**	0.934	*0.953*	0.944
D6	0.467	0.909	0.941	**0.949**	0.888	*0.947*	0.905	0.940	0.858
D7	0.285	*0.712*	0.706	0.703	0.697	0.721	0.673	**0.735**	0.699
D8	**1**	**1**	**1**	**1**	*0.999*	**1**	**1**	**1**	**1**
D9	0.710	**0.798**	0.712	0.735	0.655	*0.741*	0.657	0.723	0.354
D10	0.752	**0.949**	0.924	0.916	0.745	0.937	0.913	*0.939*	0.915
D11	0.881	0.838	*0.943*	**0.945**	0.929	0.934	0.886	0.901	0.861
D12	0.437	0.617	0.659	*0.703*	**0.715**	0.701	0.680	0.692	0.635
D13	0.612	0.631	0.698	**0.731**	0.620	0.720	0.706	*0.725*	0.697
D14	0.663	0.744	0.796	*0.807*	0.658	*0.807*	0.793	**0.811**	0.752
D15	0.429	0.604	0.649	*0.702*	0.593	0.685	0.663	0.633	**0.704**
D16	0.680	**0.772**	0.699	0.685	0.680	0.726	0.708	0.693	*0.753*

Table 3 (continued)

Data set	Base	SOMM	SMOTE	MAHAKIL	SWIM	ADASYN	DBSMOTE	MWMOTE	DIWO
D17	0.207	0.404	0.415	0.418	0.422	**0.505**	*0.453*	0.393	0.434
D18	0.949	0.950	0.947	**0.978**	*0.977*	0.958	0.941	0.949	0.965
D19	0.708	**0.921**	0.833	*0.872*	0.735	0.872	0.841	*0.872*	0.845
D20	0.569	0.589	0.630	*0.689*	**0.690**	0.687	0.666	0.662	0.649

Table 4 The best *g*-mean value obtained by synthetic over-sampling method for extreme imbalance level (1%)

Data set	Base	SOMM	SMOTE	MAHAKIL	SWIM	ADASYN	DBSMOTE	MWMOTE	DIWO
SD1	0.696	**0.941**	0.902	0.896	*0.914*	0.839	0.842	0.854	0.896
SD2	0.452	**0.942**	0.929	*0.941*	0.923	0.935	0.914	0.911	0.939
SD3	0.997	**1**	0.997	*0.999*	**1**	*0.999*	*0.999*	*0.999*	**1**
SD4	0.568	0.972	**0.977**	0.972	*0.974*	*0.974*	0.967	0.965	0.973
SD5	0.114	**0.915**	0.886	0.894	0.872	*0.904*	0.895	0.780	**0.915**
SD6	0.073	*0.904*	0.880	0.895	**0.933**	0.895	0.887	0.853	0.903
D1	*0.962*	**0.963**	0.960	0.956	**0.963**	0.952	0.940	0.959	**0.963**
D2	0.735	**0.846**	*0.844*	0.812	0.823	0.750	0.767	0.767	0.798
D3	0.645	**0.918**	0.873	0.861	*0.913*	0.720	0.701	0.611	**0.918**
D4	0.254	**0.828**	*0.821*	0.784	0.780	0.677	0.598	0.495	0.692
D5	0.289	**0.883**	0.864	*0.868*	0.863	0.701	0.637	0.453	0.751
D6	0.143	**0.849**	*0.841*	0.799	0.838	0.700	0.611	0.424	0.644
D7	0.031	0.593	0.492	0.442	**0.671**	0.538	0.459	0.453	*0.666*
D8	**1**	**1**	**1**	**1**	*0.998*	**1**	**1**	**1**	**1**
D9	0.619	**0.763**	*0.688*	0.638	0.609	0.673	0.514	0.499	0.615
D10	0.639	**0.941**	0.877	0.888	0.643	0.871	0.757	0.839	*0.892*
D11	0.828	0.811	*0.856*	0.821	**0.909**	0.769	0.752	0.755	0.756
D12	0.339	0.568	0.576	0.613	**0.664**	*0.623*	0.575	0.524	0.597
D13	0.501	0.675	*0.696*	**0.708**	0.657	0.679	0.636	0.638	0.679
D14	0.581	0.675	0.669	0.643	0.579	*0.710*	0.701	0.672	**0.716**
D15	0.350	0.557	0.583	0.539	0.552	**0.622**	0.580	0.601	*0.620*
D16	0.677	**0.765**	0.698	0.678	0.675	0.672	0.665	0.669	*0.741*

Table 4 (continued)

Data set	Base	SOMM	SMOTE	MAHAKIL	SWIM	ADASYN	DBSMOTE	MWMOTE	DIWO
D17	0.171	0.359	0.406	0.377	*0.410*	**0.463**	0.375	0.352	0.397
D18	0.816	*0.948*	0.898	0.935	**0.971**	0.894	0.899	0.872	0.945
D19	0.615	**0.851**	0.754	*0.825*	0.726	0.760	0.783	0.737	0.789
D20	0.525	0.612	0.610	0.607	**0.651**	0.633	0.633	0.642	*0.647*

Table 5 The best *g*-mean value obtained by synthetic over-sampling method for absolute imbalance level (size 6)

Data set	Base	SOMM	SMOTE	MAHAKIL	SWIM	ADASYN	DBSMOTE	MWMOTE	DIWO
SD1	0.640	**0.936**	0.859	0.851	*0.915*	0.777	0.793	0.846	0.882
SD2	0.341	**0.939**	0.920	*0.936*	0.915	0.926	0.876	0.846	0.930
SD3	0.995	**0.999**	0.995	0.995	**0.999**	0.996	0.995	0.996	*0.997*
SD4	0.410	0.968	**0.971**	0.968	0.965	*0.969*	0.954	0.953	*0.969*
SD5	0.042	**0.908**	0.882	0.889	0.867	*0.898*	0.854	0.765	**0.908**
SD6	0.024	0.891	0.874	0.881	**0.929**	0.886	0.878	0.833	*0.892*
D1	0.957	0.953	0.953	0.951	**0.962**	0.952	0.940	0.959	*0.960*
D2	0.725	**0.764**	0.722	0.638	0.751	*0.757*	0.747	0.679	0.728
D3	0.635	0.749	0.684	0.649	*0.842*	0.694	0.666	0.627	**0.883**
D4	0.235	*0.664*	0.651	0.603	**0.712**	0.646	0.571	0.437	0.520
D5	0.202	**0.709**	0.663	0.596	*0.685*	0.672	0.617	0.396	0.634
D6	0.156	*0.676*	0.672	0.618	**0.722**	*0.676*	0.584	0.400	0.632
D7	0.037	0.538	0.421	0.382	*0.635*	0.450	0.361	0.287	**0.642**
D8	**1**	**1**	**1**	**1**	*0.998*	**1**	**1**	**1**	**1**
D9	0.553	*0.612*	0.600	0.507	0.553	**0.669**	0.51	0.429	0.606
D10	0.416	**0.813**	0.647	0.544	0.716	0.641	0.531	0.636	*0.721*
D11	*0.783*	0.780	0.781	0.764	**0.888**	0.769	0.752	0.755	0.726
D12	0.337	0.590	0.579	0.557	**0.644**	0.623	0.574	*0.604*	0.574
D13	0.145	*0.529*	0.513	0.499	0.520	**0.559**	0.503	0.437	0.521
D14	0.583	**0.677**	0.666	0.644	0.584	*0.674*	0.664	0.629	0.595
D15	0.297	0.504	0.539	0.489	0.507	**0.578**	0.515	0.487	*0.558*
D16	0.675	**0.762**	0.694	0.658	0.670	0.672	0.665	0.669	*0.733*

Table 5 (continued)

Data set	Base	SOMM	SMOTE	MAHAKIL	SWIM	ADASYN	DBSMOTE	MWMOTE	DIWO
D17	0.171	0.363	0.407	0.372	*0.411*	**0.463**	0.375	0.361	0.354
D18	0.460	0.788	0.910	0.841	**0.932**	0.918	0.886	*0.919*	0.901
D19	0.791	**0.878**	0.708	*0.876*	0.735	*0.876*	0.799	0.702	0.765
D20	0.548	0.591	0.588	0.547	0.634	*0.652*	0.649	0.559	**0.709**

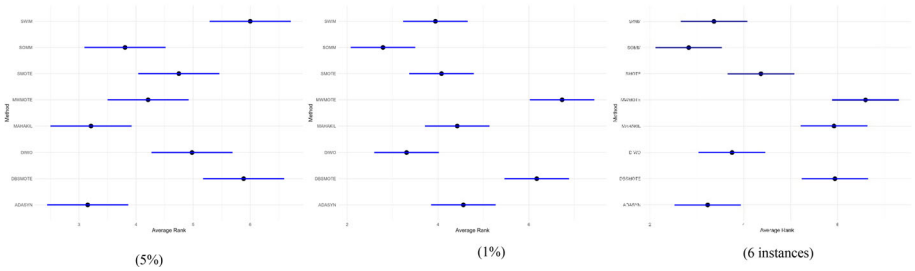


Fig. 7 Critical Difference (CD) plot

these tables, SOMM and SWIM have superior performance. SOMM performs better using k-NN, and SWIM performs better using RF and DT. Tables 16–21 (supplementary materials) provide comparative results using AUC metric.

6.2.4 Statistical testing

We compare the performance of synthetic over-sampling methods using the nonparametric statistical tests across three imbalance levels. The Friedman tests have shown the significant differences in the performance of the synthetic over-sampling methods. Figure 7 shows the CD plot for the methods across all data sets for each imbalance level. The results are in line with that is described above. SOMM performs the best for the extreme imbalance level (1%) and the absolute imbalance level (6 minority instances) with average ranking of 2.8 and 2.83, respectively. For the high imbalance level (5%), SOMM performs competitive and is amongst the first three.

6.2.5 Strengths and limitations

Considering synthetic data sets, SOMM performs the best when the majority class is bi-modal (SD2), and the minority space is non-convex (SD1 & SD5). However, SWIM is superior when minority space is convex and overlaps with the high-density region of the majority space (SD6).

By looking at results for real data sets in Tables 3, 4 and 5, we can see there are a few data sets for which SOMM does not perform best for any imbalance level in comparison with other methods. On the other hand, there are some data sets for which SOMM always performs best. In this section, we visualise two examples from each group using the first two PC plots to understand why SOMM works well or otherwise. We visualise the instances of training and test subsets from both classes to show the data spaces, the training instances used for synthetic over-sampling and the real minority space (test subset) with which synthetic instances should correspond. We also generate synthetic instances using eight methods to provide an understanding on how these methods work.

Figures 8 and 9 show the plots for data sets D15 and D18, respectively, where always a synthetic over-sampling method other than SOMM performs best. In D15, the spaces of minority and majority classes significantly overlap, but there are few minority instances out of the overlapping space (outliers). As SOMM tries to generate synthetic instances outside of the majority neighbourhood, it generates many instances close to outliers when outliers are amongst the training's minority instances. In D18, the minority space is convex. Thus,

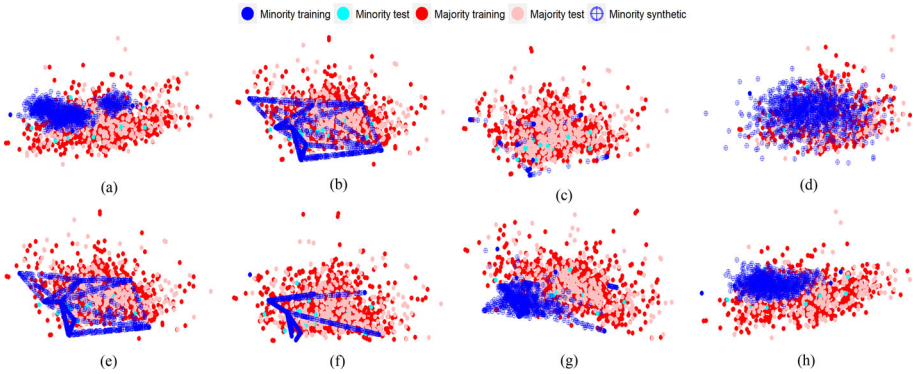


Fig. 8 PC plots for an exemplary data set D15 that SOMM does not perform the best amongst methods with synthetic generated by SOMM (**a**), SMOTE (**b**), MAHAKIL (**c**), SWIM (**d**), ADASYN (**e**), DBSMOTE (**f**), MWMOTE (**g**) and DIWO (**h**)

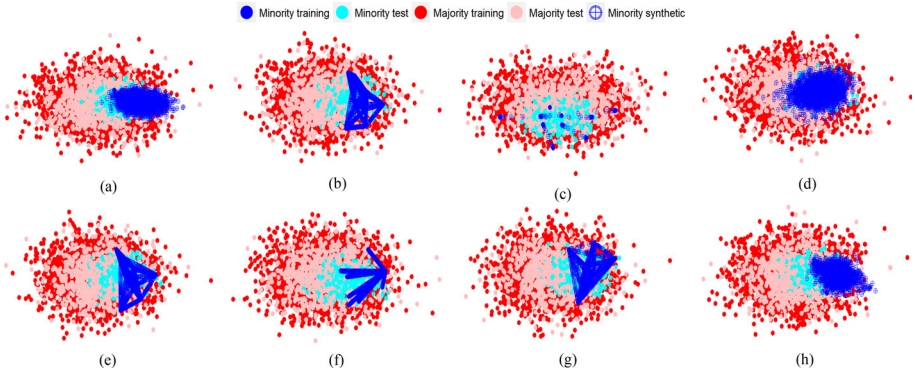


Fig. 9 PC plots for an exemplary data set D18 that SOMM does not perform the best amongst methods with synthetic generated by SOMM (**a**), SMOTE (**b**), MAHAKIL (**c**), SWIM (**d**), ADASYN (**e**), DBSMOTE (**f**), MWMOTE (**g**) and DIWO (**h**)

all methods including SOMM can perform well to generate synthetic instances inside the minority space. That is the reason MAHAKIL is the superior method when the imbalance level is high. However, another characteristic for this data set is that the distributional information of the classes is helpful to separate two classes. So, SWIM can cover the minority space by generating instances on the hyper-elliptical density contours when the number of minority instances in the training data set decreases. This is consistent with what is observed in SD6.

Figures 10 and 11 show the plots for data sets D16 and D19, respectively, where SOMM is always superior. In D16, the spaces of minority and majority classes overlap but not significantly. The minority space is bigger than the convex hull formed by the minority instances in training subset. Thus, SMOTE family and MAHAKIL fail to generate instances to cover the real space of the minority class. MAHAKIL also generates less diverse instances even within the convex hull. On the other hand, SWIM widens the minority space mistakenly as the distributional information of majority class is not informative for the minority class. However, SOMM can generate synthetic instances diversely within the real space of the

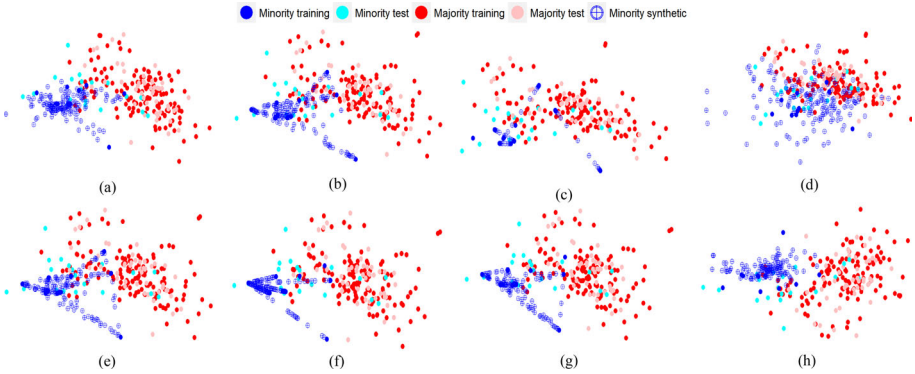


Fig. 10 PC plots for an exemplary data set D16 that SOMM performs the best amongst methods with synthetic generated by SOMM (**a**), SMOTE (**b**), MAHAKIL (**c**), SWIM (**d**), ADASYN (**e**), DBSMOTE (**f**), MWMOTE (**g**) and DIWO (**h**)

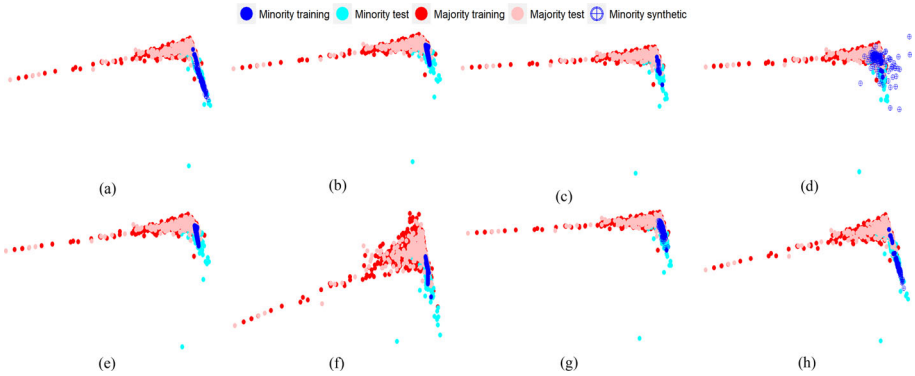


Fig. 11 PC plots for an exemplary data set D19 that SOMM performs the best amongst methods with synthetic generated by SOMM (**a**), SMOTE (**b**), MAHAKIL (**c**), SWIM (**d**), ADASYN (**e**), DBSMOTE (**f**), MWMOTE (**g**) and DIWO (**h**)

minority class. In D19, the minority space is convex, but it has a long tail. SOMM is the only method that can cover the real space of the minority class successfully. It is worth mentioning that SWIM fails to generate synthetic instances using the original data for several data sets (including D16 and D19) as the inverse matrix cannot be computed. Thus, we calculate the QR-decomposition of the majority subset to identify independent features. Then, SWIM is applied to the reduced space as instructed in [14].

We also show PC plots of D8 to provide further insights. Figure 12 shows that the two classes are completely separated and classifiers like SVM (with radial kernel) can easily distinguish between two classes even using the base model. Although ADASYN, DBSMOTE and MAHAKIL do not provide diverse coverage over the real minority class, the classification is successful. However, SWIM generates instances that are misleading for classifiers due to the tri-modal space of the majority class. It indicates that relying only on the distributional information of the majority class to generate synthetic instances is not sufficient. Figures 7,

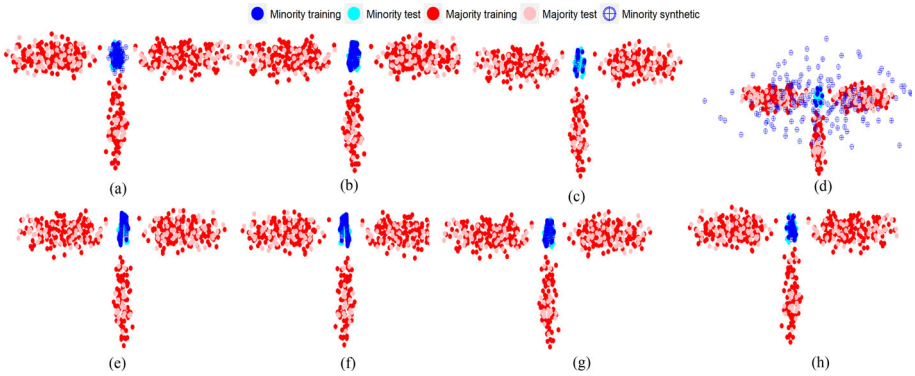


Fig. 12 PC plots for data set D8 by SOMM (**a**), SMOTE (**b**), MAHAKIL (**c**), SWIM (**d**), ADASYN (**e**), DBSMOTE (**f**), MWMOTE (**g**) and DIWO (**h**)

8 and 9 also show SOMM, SWIM and DIWO do not generate the synthetic instances using linear interpolations like SMOTE family.

7 Experiment results for multiclass imbalance problems

7.1 Experimental design

We design experiments to compare the performance of SOMM with MDO, SMOM and GMMSampling methods for multiclass imbalance problems. MDO and SMOM have been shown to be superior in comparison with the SMOTE family [38, 39]. We set the parameters for MDO as $K1 = 10$ and $K2 = 5$, and for SMOM $k1 = 12$, $k2 = 8$, $rTh = 5/8$, $nTh = 10$, $w1 = 0.2$, $w2 = 0.5$, $r1 = 1/3$ and $r2 = 0.2$. We set the parameter for SOMM as $k = 15$ as shown in Fig. 5.

We use 10 benchmark data sets (listed in Table 6) in this experimental design. These data sets have various ranges of size, dimension, number of classes and Imbalance Ratio (IR), which is calculated as (8).

$$IR = \sum_{i=1}^{C-1} \sum_{j>i} \left(\frac{N_i}{N_j} - 1 \right) \tag{8}$$

where C is the number of classes, N_i and N_j refer to the number of instances for class I and j, respectively, classes are sorted in a way that if $j > i, N_i >= N_j$.

We divide the data sets into training and testing subsets in the same way as for binary imbalanced data sets and repeat the experiments 30 times. We use the same classifiers except GLM, which is only used for binary classification, to construct classifiers on imbalanced training subset (base model) and balanced training subsets by SOMM, MDO and SMOM. We measure the performance of the classifiers on the testing subsets MG [55], which is a common measure for multiclass classification. MG is calculated as (9).

$$MG = \left(\prod_{i=1}^{C} Recall_i \right)^{\frac{1}{C}} \tag{9}$$

Table 6 Data sets for multiclass imbalance problems

Data set	Name	Class	Dimension	Size	Class description	Class distribution (%)	IR
MC-D1	Wine Quality	3	11	6497	3,4/5,6,7/8,9	4/76/20	24.8
MC-D2	Wine Quality	5	11	6497	3,4/5/6/7/8,9	4/33/44/17/3	52.0
MC-D3	Wine Red Quality	4	11	1599	3,4/5/6/7,8	4/42.5/40/13.5	25.2
MC-D4	Wine White Quality	5	11	4898	3,4/5/6/7/8,9	3.5/30/45/18/3.5	49.8
MC-D5	Vehicle	3	18	846	Opel,Saab/Bus/Van	51/26/23	2.3
MC-D6	Vowel 3	3	10	990	0/1,2,3,4,5,6,7,8,9/10	9/82/9	16.2
MC-D7	Vowel 5	5	10	990	0/1/2,3,4,5,6,7,8/9/10	9/9/64/9/9	24.4
MC-D8	Forest Cover	6	54	3000	1/2/3/4,5/6/7	34/50.5/6/2.5/3/4	93.7
MC-D9	Poker	6	10	5000	0/1/2/3/4/5,6,7,8,9	50/43/4.3/2/0.3/0.4	637.0
MC-D10	Abalone 9–18	3	9	2676	9–13/14,15/16–18	85/9/6	22.1

where *Recall* for each class is the *TPR* for that class ($\frac{TP}{TP+FN}$).

We compare the synthetic over-sampling methods and the base model using the average of MG for the best-performing classifier. We also investigate the computational time of SOMM, MDO, SMOM and GMMSampling because we have outlined the lower computation complexity as one of the advantages of SOMM over others.

7.2 Results

Table 7 presents the average of MG for the best-performing classifier of the base model, SOMM, MDO, SMOM GMMSampling (GMMS) for 10 data sets. The results show that SOMM is competitive for multiclass imbalance problems as it outperforms the base model, MDO and GMMSampling for all data sets. SOMM cannot outperform SMOM for some data sets. However, SOMM has achieved higher values for three data sets, it has the same value for one data set, and it has achieved close values for three data sets.

Table 8 presents the average runtime (in seconds) for the best-performing classifier for SOMM, MDO, SMOM and GMMSampling for 10 data sets. SOMM outperforms all MDO, SMOM and GMMSampling by achieving the lowest runtime values in all data sets. SMOM has the highest runtime values in all data sets and is on average more than 200 times slower than SOMM.

Although SOMM cannot outperform SMOM in terms of the MG metric for all data sets, the runtime complexity of SOMM is much less than SMOM. For example, SOMM can improve the base model in 166 s on average, while SMOM can do so in 69,466 s (about 19 h) on average. Overall, the experimental results validate the performance of SOMM for both binary and multiclass imbalance problems.

8 Conclusion

We have proposed a method to generate synthetic instances to alleviate the class imbalance problem in classification by over-sampling the minority class. Reviewing the limitations of existing synthetic over-sampling methods, we have defined four research questions (RQ1-4) for this study. The proposed method, SOMM, generates synthetic instances by considering the information of both majority and minority classes. This means it does not generate instances in the regions belonging to the majority class. This is particularly useful when the minority data space is non-convex, or the majority data space is not unimodal. We tested this using synthetic data sets and discussed with visualisations (addressing RQ1). SOMM generates synthetic instances diversely over the minority space. This is examined using our proposed measure (addressing RQ2) and provides an opportunity for generating synthetic instances that correspond with the real data space of the minority class and consequently are useful for constructing robust classifiers. SOMM updates the generated instances adaptable to both majority and minority data spaces (addressing RQ3). This adaptability helps the proposed method to perform well for both binary and multiclass imbalance problems.

We have examined the performance of SOMM in comparison with existing methods for binary imbalance problems using 6 synthetic data sets and 20 real data sets with high (around 5%), extreme (around 1%) and absolute (few minority instances) imbalance levels. The results show that SOMM outperforms other methods for extreme imbalance data, while remaining competitive for high and absolute imbalance data (addressing RQ4).

Table 7 The best MG value obtained by synthetic over-sampling methods for multiclass imbalance data

Data set	Base	SOMM	MDO	SMOM	GMMS	Data set	Base	SOMM	MDO	SMOM	GMMS
MC-D1	0.439	0.477	0.423	**0.547**	0.453	MC-D6	0.900	0.974	0.804	**0.981**	0.952
MC-D2	0.415	0.480	0.419	**0.513**	0.473	MC-D7	0.888	0.965	0.831	**0.975**	0.949
MC-D3	0.396	0.490	0.371	**0.492**	0.397	MC-D8	0.145	**0.494**	0.256	0.453	0.324
MC-D4	0.453	0.518	0.454	**0.525**	0.501	MC-D9	0.000	**0.067**	0.011	0.045	0.031
MC-D5	0.961	**0.969**	0.795	**0.969**	0.900	MC-D10	0.157	**0.321**	0.249	0.298	0.311

Table 8 The average of runtime in seconds by synthetic over-sampling methods

Data set	SOMM	MDO	SMOM	GMMS	Data set	SOMM	MDO	SMOM	GMMS
MC-D1	**166**	4367	69,466	6097	MC-D6	**16.7**	58.4	995.1	378
MC-D2	**225**	8301	81,345	8950	MC-D7	**29.6**	115.1	2818.5	245
MC-D3	**19.4**	478	2,134	378	MC-D8	**510.4**	3,961	97,789	7370
MC-D4	**129**	4679	52,567	5152	MC-D9	**1,140**	7,834	125,762	9967
MC-D5	**5.3**	169	858.1	201	MC-D10	**184.5**	342.9	64,901	7419

The main strengths of SOMM are that it performs effectively for both binary and multiclass imbalance problems and its effectiveness for different imbalance levels, even few minority instances. We have provided a detailed discussion of the results where SOMM outperforms other methods using PC plots. We have observed that SOMM can effectively generate diverse synthetic instances within the minority data space for real data sets. SOMM also handles the situation where the majority class does not have a unimodal distribution.

We have investigated the results where SOMM does not stand out in comparison with other methods to identify the limitations of the proposed method. We have recognised the limitation of SOMM in generating synthetic instances when the majority class is dispersed throughout the data space and the spaces of minority and majority classes overlap significantly. We suggest overcoming this limitation as future research by considering patterns and density of minority instances. SOMM does not perform significantly better in comparison with SWIM when distributional information is beneficial for generating synthetic instances. This distributional information could be considered in the future research to improve SOMM. Even though SOMM has shown potential for multiclass imbalance classification, further research is needed to extend and improve SOMM for multiclass problems as well as a one-versus-rest approach.

SOMM is beneficial for researchers/practitioners to improve the performance of their classifiers for imbalanced data with various characteristics. It can perform effectively when the number of minority instances is low, while other re-sampling methods mostly fail to improve classifiers' performance. It can work effectively for both binary and multiclass imbalance problems, whereas other re-sampling methods are mostly effective for either binary or multiclass classification tasks. In case of multiclass imbalance problems, SOMM improves classification with low time complexity.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s10115-025-02394-6.

Author contributions H.K. wrote the main manuscript text and prepared figures and tables. H.K. conceptualised the study and conducted the experiments. H.K. and U.A. revised the paper. U.A. has done the supervision. All authors reviewed the manuscript.

Funding Open Access funding enabled and organized by CAUL and its Member Institutions. No funding exists to report.

Data availability https://github.com/hadi1453/Synthetic-Over-sampling-with-the-Minority-and-Majority-classes-for-imbalance-problems/blob/main/SOMM.R.

Code availability The code is available at https://github.com/hadi1453/Synthetic-Over-sampling-with-the-Minority-and-Majority-classes-for-imbalance-problems.

Declarations

Conflict of interest The authors declare no competing interests.

Consent to participate Not applicable.

Consent for publication Not applicable.

Ethics approval Not applicable.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give

appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

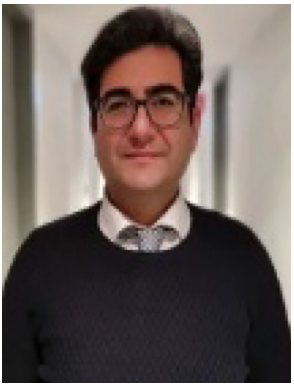
References

1. Wang S, Yao X (2013) Using class imbalance learning for software defect prediction. IEEE Trans Reliab 62(2):434–443
2. Miri Rostami S, Ahmadzadeh M (2018) Extracting predictor variables to construct breast cancer survivability model with class imbalance problem. J Data Min 6(2):263–276
3. Protopapa KL, Simpson JC, Smith NCE, Moonesinghe SR (2014) Development and validation of the surgical outcome risk tool (SORT). Br J Surg 101(13):1774–1783
4. Huang C, Li Y, Loy CC, Tang X (2020) Deep imbalanced learning for face recognition and attribute prediction. IEEE Trans Pattern Anal Mach Intell 42(11):2781–2794
5. Ji Z, Yu X, Yu Y, Pang Y, Zhang Z (2021) Semantic-guided class-imbalance learning model for zero-shot image classification. IEEE Trans Cybern
6. Zhou X, Zhang Z, Wang L, Wang P (2019) A model based on siamese neural network for online transaction fraud detection. In: International joint conference on neural networks (IJCNN), Budapest, Hungary
7. Muñoz MA, Villanova L, Baatar D, Smith-Miles K (2018) Instance spaces for machine learning classification. Mach Learn 107(1):109–147
8. Thabtah F, Hammoud S, Kamalov F, Gonsalves A (2020) Data imbalance in classification: experimental evaluation. Inf Sci 513:429–441
9. Japkowicz N, Stephen S (2002) The class imbalance problem: a systematic study. Intell Data Anal 6(5):429–449
10. Domingos P (1999) Metacost: a general method for making classifiers cost-sensitive. In: ACM SIGKDD international conference on Knowledge discovery and data mining, San Diego, USA, pp 155–164
11. Zou Q, Xie S, Lin Z, Wu M, Ju Y (2016) Finding the best classification threshold in imbalanced classification. Big Data Res 5:2–8
12. Xue J-H, Hall P (2014) Why does rebalancing class-unbalanced data improve AUC for linear discriminant analysis? IEEE Trans Pattern Anal Mach Intell 37(5):1109–1112
13. Chawla NV (2010) Data mining for imbalanced datasets: an overview. In: Maimon O, Rokach L (eds) Data mining and knowledge discovery handbook, pp 875–886, Boston, MA: Springer
14. Sharma S, Bellinger C, Krawczyk B, Zaiane O, Japkowicz N (2018) Synthetic oversampling with the majority class: a new perspective on handling extreme imbalance. In: IEEE international conference on data mining (ICDM), Singapore, pp 447–456
15. Liu A, Ghosh J, Martin C (2007) Generative oversampling for mining imbalanced datasets. In: IEEE international conference on data mining (ICDM), Omaha, USA, pp 66–72
16. Chawla NV, Bowyer KW, Hall LO, Kegelmeyer WP (2002) SMOTE: synthetic minority over-sampling technique. J Artif Intell Res 16:321–357
17. Bennin KE, Keung J, Phannachitta P, Monden A, Mensah S (2018) MAHAKIL: diversity based oversampling approach to alleviate the class imbalance issue in software defect prediction. IEEE Trans Softw Eng 44(6):534–550
18. Han H, Wang W-Y, Mao B-H (2005) Borderline-SMOTE: a new over-sampling method in imbalanced data sets learning. In: Advances in intelligent computing, Hefei, China, pp 878–887
19. Haibo H, Yang B, Garcia EA, Shutao L (2008) ADASYN: adaptive synthetic sampling approach for imbalanced learning. In: IEEE international joint conference on neural networks (IEEE World Congress on Computational Intelligence), Hong Kong, China, pp 1322–1328
20. Bunkhumpornpat C, Sinapiromsaran K, Lursinsap C (2012) DBSMOTE: density-based synthetic minority over-sampling technique. Appl Intell 36(3):664–684
21. Lim P, Goh CK, Tan KC (2017) Evolutionary cluster-based synthetic oversampling ensemble (ECO-Ensemble) for imbalance learning. IEEE Trans Cybern 47(9):2850–2861
22. Barua S, Islam MM, Yao X, Murase K (2014) MWMOTE: majority weighted minority oversampling technique for imbalanced data set learning. IEEE Trans Knowl Data Eng 26(2):405–425
23. Fernández A, Garcia S, Herrera F, Chawla NV (2018) SMOTE for learning from imbalanced data: progress and challenges, marking the 15-year anniversary. J Artif Intell Res 61:863–905

24. Dablain D, Krawczyk B, Chawla NV (2022) DeepSMOTE: fusing deep learning and SMOTE for imbalanced data. IEEE Trans Neural Netw Learn Syst. https://doi.org/10.1109/TNNLS.2021.3136503
25. Bellinger C, Corizzo R, Japkowicz N (2021) Calibrated resampling for imbalanced and long-tails in deep learning. In: International conference on discovery science, Halifax, Canada, pp 242–252
26. Choi H, Jung D, Kim S, Yoon S (2021) Imbalanced data classification via cooperative interaction between classifier and generator. IEEE Trans Neural Netw Learn Syst
27. Sutton WS (1903) The chromosomes in heredity. Biol Bull 4(5):231–250
28. Mahalanobis PC (1936) On the generalized distance in statistics. In: National Institute of Science of India
29. Bellinger C, Sharma S, Japkowicz N, Zaïane OR (2020) Framework for extreme imbalance classification: SWIM—sampling with the majority class. Knowl Inf Syst 62(3):841–866
30. Shi H, Zhang Y, Chen Y, Ji S, Dong Y (2022) Resampling algorithms based on sample concatenation for imbalance learning. Knowl-Based Syst 245:108592
31. Dai Q, Liu J-w, Yang J-p (2022) Class-imbalanced positive instances augmentation via three-line hybrid. Knowl-Based Syst 257:109902
32. Wang F, Zheng M, Hu X, Li H, Wang T, Chen F (2024) FIAO: feature information aggregation oversampling for imbalanced data classification. Appl Soft Comput 161:111774
33. Jia L, Wang Z, Sun P, Wang P (2024) HGDO: an oversampling technique based on hypergraph recognition and Gaussian distribution. Inf Sci 679:120891
34. Li J (2024) Oversampling framework based on sample subspace optimization with accelerated binary particle swarm optimization for imbalanced classification. Appl Soft Comput 162:111708
35. Li J, Li T (2023) A sample subspace optimization-based framework for addressing mislabeling in self-labeled semi-supervised classification. Appl Soft Comput 146:110687
36. Li J (2024) A self-training method based on fast binary bare-bones particle swarm optimization for semi-supervised classification. Eng Appl Artif Intell 136:108546
37. Krawczyk B, Koziarski M, Wozniak M (2020) Radial-based oversampling for multiclass imbalanced data classification. IEEE Trans Neural Netw Learn Syst 31(8):2818–2831
38. Abdi L, Hashemi S (2016) To combat multi-class imbalanced problems by means of over-sampling techniques. IEEE Trans Knowl Data Eng 28(1):238–251
39. Zhu T, Lin Y, Liu Y (2017) Synthetic minority oversampling technique for multiclass imbalance problems. Pattern Recogn 72:327–340
40. Ester M, Kriegel H-P, Sander J, Xu X (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. In: International conference on knowledge discovery and data mining, Portland, USA
41. Naglik I, Lango M (2024) GMMSampling: a new model-based, data difficulty-driven resampling method for multi-class imbalanced data. Mach Learn 113(8):5183–5202
42. Dai Q, Wang L-h, Xu K-l, Du T, Chen L-f (2024) Class-overlap detection based on heterogeneous clustering ensemble for multi-class imbalance problem. Expert Syst Appl 255:124558
43. Dai W, Ng K, Severson K, Huang W, Anderson F, Stultz C (2019) Generative oversampling with a contrastive variational autoencoder. In: IEEE international conference on data mining (ICDM), Beijing, China, pp 101–109
44. Yang Y, Khorshidi HA, Aickelin U, Nevgi A, Ekinci E (2021) On the importance of diversity in re-sampling for imbalanced data and rare events in mortality risk models. In: Australasian conference on health informatics and knowledge management (HIKM), Dunedin, New Zealand
45. Khorshidi HA, Aickelin U (2021) Constructing classifiers for imbalanced data using diversity optimisation. Inf Sci 565:1–16
46. Chen Q, Zhang Z-L, Huang W-P, Wu J, Luo X-G (2022) PF-SMOTE: a novel parameter-free SMOTE for imbalanced datasets. Neurocomputing 498:75–88
47. Yang Y, Khorshidi HA, Aickelin U (2022) Cluster-based diversity over-sampling: a density and diversity oriented synthetic over-sampling for imbalanced data. In: ICETE international conference on E-business and telecommunication networks (international joint conference on computational intelligence), Malta
48. Dai Q, Liu J-w, Shi Y-h (2023) Class-overlap undersampling based on Schur decomposition for class-imbalance problems. Expert Syst Appl 221:119735
49. Yang Y, Khorshidi HA, Aickelin U (2024) A review on over-sampling techniques in classification of multi-class imbalanced datasets: insights for medical problems. Front Digit Health 6
50. Siriseriwan W (2020) smotefamily: a collection of oversampling techniques for class imbalance problem based on SMOTE, https://cran.r-project.org/web/packages/smotefamily/index.html.
51. Cordón I, García S, Fernández A, Herrera F (2021) Imbalance: preprocessing algorithms for imbalanced datasets. https://cran.r-project.org/web/packages/imbalance/imbalance.pdf

52. Alfarrarjeh A, Kim SH, Deshmukh A, Rajan S, Lu Y, Shahabi C (2018) Spatial coverage measurement of geo-tagged visual data: a database approach. In: IEEE fourth international conference on multimedia big data (BigMM), Xi'an, China
53. Mullick SS, Datta S, Dhekane SG, Das S (2020) Appropriateness of performance indices for imbalanced data classification: an analysis. Pattern Recogn 102:107197
54. Pereira DG, Afonso A, Medeiros FM (2015) Overview of Friedman's test and post-hoc analysis. Commun Stat Simul Comput 44(10): 2636–2653
55. Sun Y, Kamel MS, Wang Y (2006) Boosting for learning multiple classes with imbalances class distribution. In: IEEE international conference on data mining (ICDM), Hong Kong, China, pp 592–602

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Hadi A. Khorshidi is a Senior Research Fellow in Cancer Health Service Research at the University of Melbourne. He has extensive research experiences in Decision-making, Optimisation, Machine Learning, Uncertainty, and Healthcare Research. He completed his PhD in Applied and Computational Mathematics at Monash University where his thesis was "System Reliability Optimisation via Uncertainty Quantification". He worked as a Research Fellow (currently an Adjunct Senior Fellow) in the School of Computing and Information Systems at the University of Melbourne. He has published more than 55 peer-reviewed journal articles and conference papers (Google citations 1500+, H-index 19). He is a chief investigator in a joint research grant-awarded project in Manchester-Melbourne Research Fund, and recipient of O&G Innovation Grant. He is an associate editor in International Journal of System Assurance Engineering and Management. He has been a member of editorial boards in International Journal of Quality and Reliability Management and The TQM Journal. He has served as a guest editor in IEEE Transactions on Evolutionary Computation (TEVC) and Information Systems and Operations Research (INFOR). He has been a member of program committee in IEEE Industrial Engineering and Engineering Management (IEEM), IEEE Symposium Series on Computational Intelligence (IEEE SSCI) and IEEE International Joint Conference on Neural Network (IJCNN) Conferences.



Uwe Aickelin is an internationally renowned academic in the fields of Artificial Intelligence, Optimisation and Data Mining and since 2018 the Head of the School of Computing and Information Systems at the University of Melbourne. His specific expertise is in the modelling stages of decision problems with a focus on robust methods to overcome uncertainty. Typical application areas of his work are Decision Support and Optimisation in Health Informatics and Security. Professor Aickelin is the Specialty Chief Editor of Frontiers in Digital Health – Health Informatics and has authored over 250 papers in leading international journals and conferences, participated in over 100 international conference programme committees and completed over 25 PhD supervisions.